```
expjmpinstr:
        ldr     pc, [pc, #24]
```

The last statement of the file is the relative jump instruction. We use the assembler to create the instruction, which is both easier to read and less prone to errors than defining a constant.

12.7 Assignment Of Device Interrupt Vector Numbers

The positions of exceptions in an exception vector are chosen when a system is designed and never change. For example, we said that the Galileo hardware is built so an illegal instruction error always raises exception 5. On the BeagleBone Black, an illegal instruction raises exception 6. However, IRQ values cannot be preassigned unless the set of devices is fixed when the hardware is built (e.g., an SoC that has three devices). To understand why, observe that most computer systems allow an owner to purchase and install new device hardware. To accommodate an arbitrary set of devices, three basic approaches have been used for IRQ assignment:

- Manual device configuration
- Automated assignment during bootstrap
- Dynamic assignment for pluggable devices

Manual device configuration. On early hardware, a human had to assign a unique IRQ to each device before the device was connected to a computer. Typically, the assignment was made using switches or wire jumpers on the device circuit board. Once an assignment was made to the hardware, the operating system had to be configured to match the hardware. Manual assignment had the problems of being tedious and error-prone — if a human accidentally assigned the same IRQ to two different devices, or the vector number configured into the operating system did not match the IRQ value configured into device hardware, devices would fail to operate correctly.

Automated assignment during bootstrap. As bus hardware became more sophisticated, techniques were developed that automated interrupt vector assignments. Automated assignment requires *programmable* devices. That is, the operating system uses the bus to find devices that are attached to the bus, and assigns each device an IRQ. In essence, programmable devices allow the paradigm to be reversed: instead of assigning an IRQ to a device and then configuring the operating system to match the assignment, programmable devices allow the operating system to choose an IRQ and then assign the number to the device. Because the operating system handles assignment when a computer boots, the automated approach eliminates human error, and makes it possible for users to attach new hardware to their computer without understanding IRQ assignment.

Dynamic assignment for pluggable devices. A final approach is used to accommodate devices that can be plugged into a computer while the operating system is running.